Course material is not allowed during the exam.
Try to keep your answers precise and short. You will not get extra points by giving very long answers or by writing down what you know instead of what is asked.
Just answer the question that is asked.
Take 10 seconds to think about what you are going to write before writing it.
1. a) Explain the dynamic taint tracking technique (also known as just tainting) in a few words.

b) Give an example of its use to protect privacy.

c) Give one important limitation of dynamic taint tracking.

2. Saltzer and Schroder proposed “8 principles of secure systems”. Explain 2 of them.

3. Check the boxes to say whether the assertions below are valid or not (good answer 1, bad answer -1, result for this question between 0 and 5):

<table>
<thead>
<tr>
<th>Assertion</th>
<th>True</th>
<th>False</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASLR (Address Space Layout Randomisation) makes overwriting the return address impossible.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Return Oriented Programming helps to bypass Stack Canaries.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stack Overflows and Stack-Based Buffer Overflows are completely different problems and are both exploitable.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Buffer Overflows are exploitable only if the buffer is allocated on the stack.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Heap-Spraying allows to build reliable exploits in the presence of NX, ASLR and Canaries.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Null pointer dereference bugs are exploitable, under uncommon conditions.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4. What is ingress filtering, what kind of attacks ingress filtering can help to prevent.

5. In wireless security, what is the name of the general technique to attack availability, give one example of its use.

6. Explain in a few words the principle of the packets in packets attack.

7. What is secure boot? Give one limitation of secure boot (i.e. possible attack)?
8. “Frankenmalware” (as from Frankenstein+malware) were recently reported to be found in the “wild”. Those are worms that were infected by a virus, they then propagate together from host to host.
   a) How this can happen?

   b) Why, by definition, the opposite (worm that infects a virus) is impossible?

   c) What is the main risk with those?

9. What are string format bugs? Why are they dangerous?
10. This is a recent advisory about “sudo”:

Phenoelit Advisory <wir-haben-auch-mal-was-gefunden #0815 ++--++>

[ Authors ]
joernchen <joernchen () phenoelit de>
Phenoelit Group (http://www.phenoelit.de)

[ Affected Products ]
sudo 1.8.0 - 1.8.3p1 (http://sudo.ws)

[ Vendor communication ]
2012-01-24 Send vulnerability details to sudo maintainer
2012-01-24 Maintainer is embarrassed
2012-01-27 Asking maintainer how the fixing goes
2012-01-27 Maintainer responds with a patch and a release date
of 2012-01-30 for the patched sudo and advisory
2012-01-30 Release of this advisory

[ Description ]
Observe src/sudo.c:
(for the exam purpose the code has been simplified a lot for
certainty)

void
sudo_debug(const char *fmt, int i)
{
    char fmt2[LARGEENOUGH];

    sprintf(fmt2, "%s: %s \n", getprogname(), fmt);
    printf(fmt2, i);
}

Here getprogname() is argv[0] and by this user controlled. So
argv[0] goes to fmt2 which then gets printf()ed. The
result is a Format String vulnerability.

[...] Some background:
- int sprintf(char *str, const char *format, ...);
    Writes the formatted string in string “str”, for the exam we assume that str is large enough to
    hold the resulting string (really, I don't want to see that in your response)
- From the sudo man page: ”sudo allows a permitted user to execute a command as the
  superuser or another user, as specified in the sudoers file.”
Questions:

a) Why a bug in “sudo” a very bad thing?

b) Explain, very briefly, how you would exploit the format string bug?